WEST

☐ Generate Collection | Print

L11: Entry 43 of 54 File: USPT Aug 3, 1999

DOCUMENT-IDENTIFIER: US 5933816 A
TITLE: System and method for delivering financial services

Abstract Text (1):
A delivery system and method allow a financial institution to provide financial services to a plurality of remote devices, such as personal computers, personal data assistants, and screen phones. In addition to providing services to these remote devices, the system and method provide services to automatic teller machines (ATMs), external service providers, and internally within the financial institution to staff terminals and to the individual branches of the financial institution. The delivery of financial services is not limited to any particular network but rather may be provided through dial-in access, Internet access, on-line service provider access, or other types of delivery networks. The system is comprised of a set of reusable global components which are modular and are organized into services sets. By separating the components of the system into independent components, the system and method can be developed and tested on a component level rather than the entire system level, thereby substantially reducing the development and maintenance cycle time. The system and method operate in sessions and, for instance, employ a dialog component for gathering information from a customer, a rule broker component for providing answers to the various legal and regulatory rules in a particular country, a language man component for selecting appropriate language, a transaction executor component for performing transactions, and a presentation manager component for formatting outputs to the customer. The system and method provide state-of-the art interfaces with interface components and support legacy applications with legacy app bridge components.

Brief Summary Text (7):
This simplified model of banking, while still in existence, has been greatly expanded. In addition to the bank tellers, banks provide Automated Teller Machines (ATMs) so that customers can perform transactions at literally any hour of the day. The locations of the ATMs are not limited simply at the bank's branch locations but can be found, for instance, in shopping malls, airports, grocery stores, hotels, and building lobbies. Since the ATMs must access the books of the banks to allow customers to perform their transactions, banks had to provide an interface between the ATMs and the bank's internal computer system that allows the ATMs limited but secure access to the bank's books.

Brief Summary Text (8):
The model for providing financial services has been expanded even further to enable home banking. With home banking, a customer can access his or her personal account and perform transactions, such as bill paying, money transfers, etc., in the convenience of one's home through the customer's personal computer. To enable home banking, banks had to provide an interface between the bank's internal computer system and the customer's personal computers to allow limited and secure access to the bank's books. Due to the differences between ATMs and personal computers, the interface for the personal computers is typically separate and distinct from the interface provided for the ATMs.

Brief Summary Text (13):
Another difficulty facing a bank entering another country is that the bank must, in effect, create a new computer system for each country that it enters. Each country has its own unique regulatory and legal environment which dictates the manner in which financial services must be performed. The bank cannot simply duplicate a computer system operating in another country but rather must specially tailor each computer system to the regulatory and legal environment of that country. This extra amount of effort required to shape a computer system according to the rules and laws of its home country consumes more of the bank's valuable resources.

Brief Summary Text (21):
The system and method operate in sessions with a session bubble instantiated for each session with a remote device. After receiving an initial contact with a remote device, a session controller instantiates a session component to manage resources for the session bubble. The session component, in turn, instantiates a number of components for the session, such as a welcome mat component, front door man component, rule broker component, and acquirier component. The welcome mat component sends a logon message to the remote device and instantiates a profile transaction executor component to authenticate a customer. A navigation shell component notifies the remote device of the list of available functions, such as cash withdrawal or bill payment, and instantiates a mini-app dialog component based on the function selected through the remote device. To coordinate communications with the plural sessions that may occur simultaneously, a touch point interface component routes incoming messages from remote devices to the appropriate session bubbles and a back door man component coordinates messaging between the various sessions and an external service provider.

Brief Summary Text (22):
The system and method employ a rule broker component that other components within the system may query to obtain an answer to any question that might arise. The rule broker component registers rule authorities as the answerers of questions and directs the querying component to the rule authority for the answer. The rules within the system and method may be modified independently from other application components. With the rule broker, regulatory or business rules can be easily added, changed, or modified with only a minimal impact on overall operations.

Detailed Description Text (5):
With reference to FIG. 2, a delivery system 12 according to a preferred embodiment of the invention comprises plural sets of service components. These sets of service components include a touch point and display set 30, a touch point interface services set 40, and a touch point services set 50. In general, the touch point and display set 30 provides actual customer display and input facility and the touch point interface services set 40 provides an interface to the touch point services set 50. The touch point services set 50 provides presentation mapping and front door security for the delivery system 12. The delivery system 12 also includes a peripheral device services set 60 providing peripheral device interface and management services. A system services set 70 provides logging, event brokering, service registry and crypto services and a dialog services set 80 provides welcoming, navigation shell and application specific dialogs. A transaction services set 90 provides transaction coordination and ESP message formatting and an external service provider interface services set 100 provides message sequencing and ESP interface protocols. A customer services set 110 provides customer identification, relationship, account, acquirer, and issuer services and a business services set 120 provides rule brokering and language, services. A session services set 130 provides session start up and session and delivery vehicle context.

Detailed Description Text (23):
The event broker component 73 provides a way for a business to do specialized processing of events for the purpose of monitoring and acting upon activities in a server. Local business provided components can register with the event broker component 73 to receive specified events. The event broker component 73 evaluates filtering rules associated with events and then calls the registered component as a result of a rule succeeding. The event broker component 73, for example, could decide when to send notifications to a system management system.

Detailed Description Text (28):
The test manager component 78 manages the testing and tracing of components in the system 12. The test manager component 78 collects information from the various components in the system 12 by wiring itself into them during component creation. Then, the components that have been wired for test report method entries and exits to the test manager component 78 during their operation. The configuration of which components are under test or trace can be driven by scripts or by an on-line test management user interface. The test manager component 78 records information reported by the components under test in a log or it can report the test results to the tester through the test management user interface. The test manager component 78 therefore knows which components are under trace and test and wires new components for tracing and testing.

Detailed Description Text (31):
The welcome mat component 81 outputs the initial welcome page to the customer and collects customer identity and preference information. After determining the issuer of the customer ID and possibly authenticating the customer, the welcome mat component 81

instantiates several customer services objects to hold information about the customer and then starts a navigation shell component 82 which carries out the next level of dialog with the customer. The welcome mat component 81 establishes connection sessions with a back door man component 101 in the ESP interface services set 100 as needed by a session. The welcome mat component 81 also acquires devices needed by the session and creates a scam transaction executor to handle unsolicited scam events from a host. The welcome mat component 81 presents an out of service or welcome page, enables a card reader, and waits for card read events. If the card event is an administration card, the welcome mat component 81 instantiates an administrative welcome mat component. The welcome mat component 81 collects various information from the customer including language choice and other preferences, such as navigation style. The welcome mat component 81 also collects customer ID information, such as CIN/PIN and public key certificate, in a manner consistent with the customer remote device and mode of access, such as dial-in or Internet. The welcome mat component 81 handles retries if errors occur on customer identity input, for instance by re-reading a card, and asks customer ID component 111 for issuer. The welcome mat component 81 instantiates a profile transaction executor component 91 to authenticate the customer and get the customer's relationships or customer profile. This process typically involves interactions with the issuer external service provider, but may alternatively be performed locally based on information in a SmartCard. The transaction executor component 91 instantiated by the welcome mat component 81 will instantiate the following customer service components: customer ID component 111, customer relationship component 113, account component 115, and issuer component 112. The welcome mat component 81 will also initialize legacy app bridge components 84, and start a navigation shell component 82 based on delivery capabilities, acquirer rules, and customer preferences.

Detailed Description Text (33):
The welcome mat component 81 may do four things for customer authentication based on acquirer rules and the type of customer ID, such as public key certificate, ATM card, credit card, on-us, or off-us. The welcome mat component 81 may provide immediate local authentication using public key certificates or may provide immediate authentication with the issuer, waiting for a response. The welcome mat component 81 may also provide background authentication with the issuer while going on to the navigation shell component 82 or may defer authentication to the first transaction. With deferred authentication, the welcome mat component 81 may need to instantiate a default customer relationship component 113 and a default set of product types, such as checking, savings, or credit card. If a rule broker component 121 does not have a registered issuer for the card/CIN prefix number, a customer ID component 111 is instantiated and marked invalid, further authentication of the customer is skipped, and a navigation shell component 82 for invalid customers is started. Invalid customers may still be allowed to use certain information only in mini-app dialogs.

Detailed Description Text (34):
The navigation shell component 82 informs the customer of the range of mini-apps that are available and provides top level navigation across these applications. The navigation shell component 82 assigns a frame space within which a mini-app runs. To support complex grouping of functions or a variety of navigation styles, the navigation shell component 82 may contain shells within shells. The navigation shell components 82 available for selection by a customer include linear, which guides customers through detailed question and answer steps; nonlinear broad branching, such as pull-down menus; preferred, such as customer specified short cuts; or query, which may include a search engine or natural language searching capabilities. The navigation shell component 82 obtains lists of possible services available from services registry component 74, checks rules to see what services are actually available in the current system context, and makes the customer aware of the range of mini-apps available. The range of mini-apps available will be based on the customer's relationship, the issuer/acquirer rules, and the set of dynamically registered mini-apps. The mini-apps may be organized and identified by the navigation shell component 82 with names, icons, or any other type of representation. The navigation shell component 82 instantiates additional navigation shell components 82 as necessary and instantiates mini-app dialog component 83 as requested by the customer. The navigation shell component 82 supports switching between concurrently active mini-app dialogs and, at the end of a session, instantiates and calls end of session mini-app. The delivery system 12 preferably supports the customer leaving a mini-app to enter the navigation shell component 82 and to start another mini-app, while leaving the former mini-app suspended in its current context state. The customer can later exit from the new mini-app and go back to the former mini-app or can switch between multiple concurrently active mini-apps. In an environment where the screen has imbedded frames, a main navigation shell component 82 may, for example, invoke one or more sub shell components 82 to control individual

frames.

Detailed Description Text (35):
The mini-app dialog component 83 manages the dialog with a customer for a specific business function in a specific dialog style. The mini-app dialog component 83, for instance, may manage the business functions of transferring funds or bill payment in the styles of question and answer or forms. The mini-app dialog component 83 presents information and choices to the customer and collects and validates customer inputs. The mini-app dialog component 83 is responsible for the content of information on pages and the flow of the customer interaction, but preferably not the style and layout of the presentation. The mini-app dialog component 83 may comprise several different mini-app dialog components 83 with different dialog styles for the same business function. The mini-app dialog components 83 may support different modes of the customer entering information, such as guiding the customer through detailed question and answer steps or forms with multiple input fields. After collecting the necessary customer inputs for a particular business function, the mini-app dialog component 83 uses a transaction executor component 91 to carry out the function by doing transactions with external service providers and operating peripheral devices, such as a cash dispenser or depositor. The mini-app dialog component 83 implements the customer-visible control flow for a particular function in a specific dialog style. The flow may be tailored based on the customer relationship and on various countries/business rules. The mini-app dialog component 83 uses a language man component 122 within the business services set 120 to do translation of phrases into target languages for display or print. The mini-app dialog component 83 assembles phrases and formatted data into pages, for display or print, with each page constructed in a canonical format by setting properties of named objects within named templates. The mini-app dialog component 83 sends pages to the presentation manager component 52 which handles the final style and layout for the specific remote device. The mini-app dialog component 83 collects customer inputs and validates customer inputs using business rules. Validation, for instance, includes basic field validations as well as cross-field validations. The mini-app dialog component 83 instantiates and calls transaction executor components 91 to do transactions with external service providers and also operates remote devices, such as a cash dispenser or a depositor, needed by the business function. The mini-app dialog component 83 queues transaction data for printed record and increments transaction counters in the instrumentation component 76. A mini-app dialog component 83 may, for instance, use separate mini-app dialog subcomponents 83 to do some parts of the dialog that may be common to several business functions, such as PIN entry, account resolution, and entering currency amount.

Detailed Description Text (36):
The legacy app bridge component 84 is a bridge that enables a legacy application set to operate in the delivery system 12. The legacy app bridge component 84 translates data between customer and business services objects in the delivery system 12 in the form that data is stored in the legacy applications. A different legacy app bridge component 84 may exist for each type of legacy application set, such as USCAT, AsiaCAT, LatinCAT, and EuroCAT. On entrance to a legacy application, the legacy app bridge component 84 obtains data from the session services set 130 and customer services set 110 and translates the data into the global data structures needed by the legacy application. On exit from a legacy application, the legacy app bridge component 84 takes modified data from the legacy structures and puts the data back to the customer services set 110 within the delivery system 12. The legacy app bridge component 84 translates legacy pages into the canonical page structures needed by the presentation manager component 52 and interfaces with the back door man component 101 to send messages to external service providers. The legacy app bridge component 84 also interfaces with the logger for logging errors and transactions. During initialization of the legacy app bridge component 84, the rule broker component 121 and various rule authorities, primarily acquirer and issuer, may need to be interrogated to obtain data needed to populate static tables used by the legacy applications for processing rules. Depending upon the extent of migration, the legacy app bridge component 84 may have several different relationships between it and the navigation shell component 82. For instance, the navigation shell component 82 may provide the top level navigation across the new mini-app dialog component 83 as well as the individual legacy app bridge component 84. For some card types and issuers, the navigation shell component 82 may be faceless and all business functionality is provided by the legacy apps. In this alternative, top level navigation may be provided within the legacy applications. For CAT applications, one of a pool of CAT/TAFE runtimes will be assigned to a session at start-up. The legacy applications will be assigned a frame space within which the navigation shell component 82 "plays" its applications. Individual CAT level 3 functions will be individually registered and exposed. The navigation shell component 82 supports

exposing CAT level 3 functions without the need to traverse the existing level 2 menu structure.

Detailed Description Text (45):
The customer services set 110 provides a category of services that includes all information specific to the customer who initiates a session. All information related to identifying the customer, the issuing business of the customer, the customer's profile, and all the customer's accounts are the component objects included within this category of services. The customer services set 110 includes a customer ID component 111, an issuer component 112, a customer relationship component 113, an acquirer component 114, and an account component 115.

Detailed Description Text (47):
The issuer component 112 represents the issuing business for the customer-ID information that was used to start a session. The issuer component 112 is the rule authority for all general, issuer related, non mini-app specific business rules. The issuer component 112 supports query of issuer information and supports answering questions about general issuer business rules. The issuer component 112 has information about the issuer of customer's identity, for instance, business code, financial institution identifier, and issuer type, such as bank card, credit card, or other third party card. The issuer component 112 knows the PIN length supported and the issuer country and ISO currency code for the issuer default currency. The issuer component 112 has a list of customer relationships for the issuer and a list of accounts for the issuer. The issuer component 112 also knows the products and services supported and the transaction and product limits. The issuer component 112 is informed of the issuer's presentation rules, such as data, format, and account number masking, and the issuer's local rules, such as collect call support, currency, and product names. The issuer component 112 also knows the issuer's servicer-ESP communication rules, for instance, profile message support, the languages supported, and the navigation schemes supported. The issuer component 112 knows when or how to authenticate customer, such as by local validation of public key certificate, immediate to issuer, background to issuer, or delayed to first transaction.

Detailed Description Text (49):
The acquirer component 114 contains information and answers about the acquirer. The acquirer component 114 represents the acquiring business for a session and is the rule authority for business rules that are acquirer related, but not mini-app specific. For rules that are acquirer related and mini-app specific, separate rule authorities may be registered as part as a dynamic installation of a mini-application. The acquirer component 114 supports query of acquirer information and processes certain specific rules associated with the acquirer. The acquirer component 114 knows information about acquiring business for a session, for instance a financial institution identifier and business code, and knows the country or region of acquirer.

Detailed Description Text (50):
The account component 115 contains information and can answer questions about a particular account. Each individual account preferably has only one account component 115 with the account details and rules varying for the particular individual account. The account component 115 supports query of account information and supports update of account information. The account component 115 knows the business owning the account, the category of the account, and the product type and subproduct type of the account. The account component 115 also knows the fund family code and fund code, the category code, the account name, account number, and account details, such as currency code, balances, and terms. The account component 115 has information on the functional privileges and limitations and also information on associated link accounts. The individual accounts may be customer owned or payee accounts that can be the target of a transfer or bill payment.

Detailed Description Text (52):
The business services set 120 provides formal mechanisms for dealing with business rules, language support, and acquirer services. The business services set 120 includes a rule broker component 121 and a language man component 122.

Detailed Description Text (53):
The rule broker component 121 formalizes a mechanism for dealing with business rules that have conventionally been ad hoc. The rule broker component 121 is a central registry for all business questions. Other components within a session address named business questions to the rule broker component 121. The rule broker component 121 routes the question to the rule authority or authorities that have registered for a

rule. By having a separable rule authority for each mini-app specific business rule, new rules can be added independently without affecting the rest of the delivery system 12. The rule broker component 121 supports the concept of overrides, which allow the dynamic registration of a new rule authority when changes to business rules are necessary. The rule broker component 121 may either answer questions directly or route questions to another component, such as an account component 115 or the issuer component 112. The rule broker component 121 is also responsible for interfacing into rules databases and knows what component will answer each question.

Detailed Description Text (57):
The session component 132 manages the resources associated with this session. The session component 132 brings up some initial session resources and is the registry for the brought up session components. The session component 132 also knows certain session context information as well as all assigned session resources and services. The session component 132 instantiates and initializes the following resources when a session is created and deletes them when the session is terminated: delivery capabilities component 133, rule broker component 121, front door man component 51, presentation manager component 52, acquirer component 114, language man component 122, and welcome mat component 81. The session component 132 sends touch point attached notification to each of the components and supports registration of additional session components that need to be accessed globally by the session. The session component 132 recovers resources when a session abnormally terminates and logs significant session events, such as start or end of session and session errors. The session component 132 has session initiation information including the session ID and the start of session time. The session component 132 also has the handles for linking to many other session components and knows which navigation shell components 82 and mini-app dialog components 83 are active. The session component 132 also knows the reason for the end of a session.

Detailed Description Text (62):
At step E6, the session component 132 instantiates the front door man component 51. The session component 132 instantiates the presentation manager component 52 at step E6 and instantiates the presentation manager component 52 at step E7. At step E8, the session component 132 instantiates the rule broker component 121 and at step E9 instantiates the language man component 122. At step E11, the session component 132 instantiates the acquirer component 114 and at step E11 instantiates the welcome mat component 81.

Detailed Description Text (64):
As reflected in steps E1 through E14, a customer can access the delivery system 12 with any type of remote device. In response, the delivery system 12 will create a session bubble specific for that customer. This session bubble will preferably have a session component 132, a delivery capabilities component 133, a session device manager component 63, a rule broker component 121, a welcome mat component 81, a front door man component 51, as well as various other components dedicated for that particular session. Through the presentation manager component 52, front door man component 51, touch point interface component 41 and touch point and display component 31, the delivery system 12 can format messages to any type of remote device and can custom tailor this message according to the desires of a particular customer. The delivery system 12 is also capable of providing uniformity across the various remote devices so that the customer is presented with a consistent and familiar interface regardless of the remote device used.

Detailed Description Text (67):
At step E25, the welcome mat component 81 asks the rule broker component 121 who is the issuer based on the CIN. The welcome mat component 81, in turn, instantiates the customer ID component 111 at step E26 and instantiates the issuer component 112 at step E27. At step E28, the welcome mat component 81 instantiates the profile transaction executor component 91 for authenticating the customer and then passes the CIN and encrypted PIN to the transaction executor component 91. At step E29, the transaction executor component 91 formats a reply message and sends the message to the host through the back door man component 101. At step E30, the back door man component 101 adds a universal message sequence and, at step E31, the external service provider interface component 102 provides protocol gateway to the external service provider 22.

Detailed Description Text (70):
The selection of a mini-app will now be described with reference to FIGS. 7A and 7B and FIGS. 8A and 8B. At step E41, the customer selects a mini-app with the touch point and display component 31 and the request is sent into the delivery system 12. At step E42, the presentation manager component 52 demultiplexes the request based on mime-type and

URL and sends the request to the navigation shell component 82. A step E43, the navigation shell component 82 instantiates the appropriate mini-app dialog component 83. At step E44, the mini-app dialog component 83 returns choices to the customer. At step E45, a back and forth dialog occurs between the customer and the mini-app dialog component 83 until all information is collected for a function. During this step, the mini-app dialog component 83 directs business rule questions to the rule broker component 121 for resolution during the dialog.

Detailed Description Text (74):
The presentation manager component 52, using delivery vehicles specific named templates, is responsible for style and mapping to the encoding language of the target device. The presentation manager component 52 takes the app stream received from the mini-app dialog component 83 and, based on delivery vehicle specific templates, merges the data based on mapping rules and produces the final token stream that is sent to the touch point and display component 31. A one to one mapping exists between canonical templates that mini-app dialog components 83 reference and delivery vehicle specific templates that the presentation manager component 52 uses. Delivery vehicle specific templates include specific information on the layout, colors, and mapping of individual objects. A set of emerging standards from Microsoft and WC3 on advance style sheets, including style sheets that allow precise X, Y positioning of objects may be used as part of the templating mechanism.

Detailed Description Text (84):
A fundamental advantage of the delivery system 12 is the independence of one mini-app dialog component 83 from another. The delivery system 12 provides a safe environment for the dynamic insertion and registration of new mini-apps with their navigation shell components 82. The delivery system 12 can introduce a new mini-app dialog component 83 so as to require a complete testing cycle only on the introduced application and not a regression test of the entire delivery system 12. The mini-app dialog components 83 are therefore preferably packaged as separate entities. For instance, a mini-app dialog component 83 may include an executable (.EXE) for the mini-app dialog component 83 including the transaction executor component 91 as a DLL or OLE object. The mini-app dialog component 83 also includes a rule file, including all new rule entries to be registered with the rule broker component 121. Also, when appropriate, the mini-app dialog component 83 includes a rules engine file per rule for any rules that can be interpreted by the general purpose rules engine and a rule database file per rule that supplies any needed data to support mini-app specific rule authorities. The mini-app dialog component 83 also includes a language file including all mini-app specific language phrases needed and, when appropriate, a template file containing all mini-app specific templates.

Detailed Description Text (87):
To start a NetCAT session, the traveling customer dips his or her card at "foreign" CAT 16 and a session bubble starts up normally at the CAT 16. When the welcome mat component 81 determines that this customer is off-region, the welcome mat component 81 makes a connection to the appropriate regional NetCAT server 200. The welcome mat component 81 on the CAT 16 communicates with the session controller component 131 on the NetCAT server 200 to start up a session. The welcome mat component 81 on the NetCAT server 200, after given card parameters upon start up, instantiates the customer ID component 111 and issuer components 112 on the NetCAT server 200. After NetCAT server 200 authenticates the customer, with its own external service provider, the NetCAT server 200 starts up a navigation shell component 82 on the NetCAT server 200. The CAT 16 exposes/copies certain of its components to the NetCAT server 200 for its use. The CAT 16, for instance, exposes the session component 132 , the acquirer component 114, the delivery capabilities component 133, the front door man component 51, the peripheral device manager component 62, the transaction executor components 91, and the welcome mat component 81. The NetCAT server 200 uses these components for business rule inquiries, for delivery to CAT screen, for operation of the CAT peripherals, and for inquiry about the capabilities of the hosting CAT, such as fonts supported and pictographic printing. An example of a CAT 16 is shown in FIG. 10 with the exposed components marked with a black dot.

Detailed Description Text (89):
The delivery system 12 supports an orderly migration of CAT functionality from implementation with AGS applications to implementation with service components on all platforms where AGS is used to delivery CAT look-and-feel functionality. An example of the interaction between CAT AGS applications and service components will be described with reference to FIG. 11. The AGS applications are executed within an instance of a TAFE process, the legacy run time AGS driver and associated functionality, and share a

single persistent global data store. At the time a CAT application is invoked, session context is completely represented by the current state of the persistent global data and the content of the Exit message TAFE passes to the application. If this context can be instantiated by alternate means, then the business/customer functionality normally performed by the AGS level one and level two applications need not be performed before running a level three transaction application. At a high level, pretransaction session context is imported to the TAFE and a level three application is invoked with Exit message. After a return from level three application with Exit message, post transaction session context is exported from TAFE. For the case of a complete session performed in AGS, the interaction includes importing prelanguage selection session context to TAFE, invoking level two application with Exit message, returning from level two application with Exit message, and exporting post end-of-session context from TAFE.

Detailed Description Text (93):
VII. Rule Broker

Detailed Description Text (94):
One advantage of the delivery system 12 is the separation of individually-installable business rules from the code embodied in the transactions specific components. Application components needing answers to rule questions asks the rule broker component 121 without knowing any details about how the rules are encoded and answered. The rule broker component 121 routes the question to the appropriate component which can supply an answer. Components which supply rule answers may be installed independently of components which ask the rule questions. In addition, any data used by a rule "answerer" may be installed or replaced independently from components which use that data to determine answers to rule questions.

Detailed Description Text (95):
In general, a business rule is a statement of policy driven by business or regulatory needs, which defines context specific behavior of the applications. A business rule in the delivery system 12 may comprise any statement policy driven by business or regulatory needs, which defines context-specific behavior of an application. Business rules are discrete items which may be modified independently from other application components. Examples of business rules are choosing dispense amounts to display, maximum PIC retries, assignment of product types to summary categories, assignment of product types to product categories, and the number of account digits on print records. On average, fifty to one hundred business rules may exist per region with most of the rules being issuer rules and a fewer number of acquirer rules.

Detailed Description Text (96):
The rule broker component 121 is a single entity which components of the delivery system 12 may access to obtain answers to the business rule questions which affect application processing. The rule broker component 121 receives rule registration requests, registers rules in a rule registry, receives rule queries and routes them to the registered provider for that rule. The rule broker component 121 provides a mechanism for rule authorities to register themselves as answers for particular rule questions. When application components query the rule broker for a particular rule, the rule broker component 121 routes the query to the appropriate rule authority or to the rule engine. The rule broker component 121 itself is not aware of the actual semantics of any of the rules. In the preferred embodiment, the rule broker component 121 is used by the mini-app dialog components 83, the transaction executor components 91, the presentation manager component 52, the navigation shell component 82, the welcome mat component 81, and the legacy app bridge component 84. Although the delivery system 12 preferably includes the rule broker component 121, certain components within the delivery system 12 can be direct answerers of questions when appropriate.

Detailed Description Text (97):
The rule authority is a component which can answer rule questions. Components within the delivery system 12 act in the role of a rule authority component if they register themselves with the rule broker component 121 as the answerer of a named rule. For instance, the issuer component 112, the acquirer component 114, and the delivery capabilities component 133 may each be a rule authority. The rule authority components register rules with the rule broker component 121 and provide answers for the rules that they have registered. The rule authority components may access separately installable data to answer rule questions and this data may be separate from the rule registry information used by the rule broker component 121 and the rule engine.

Detailed Description Text (98):

The rule engine is a general rule interpreter. The rule engine can answer a rule query based on parameters passed in the query and some interpretable rule data in the rule database. Unlike rule authorities, the rule engine has no specific knowledge of rules or applications. The rule engine determines answers for rules and is used by the rule broker component 121 and calls the rule registry.

Detailed Description Text (99):
In operation, each rule registered with the rule broker will have a unique name which includes a version identifier. The name will be passed separately from other parameters in a rule query. All rule query parameters aside from the name will be passed in a self-defining way, for instance, a rule query may contain a name, type, and value for each parameter. Each rule registered with the rule broker will exist as an independent record in a rule registry. A 15 rule in the rule registry will be defined as either data, such as an encoded string, which can be interpreted by a general rules engine or a rule authority which is registered to answer a rule. A component's registration of a rule may override a previous component's registration for that same rule. Each registered rule will define the expected type of parameters to be passed and rules can be dynamically added to the rule registry independently of all other rules.

Detailed Description Text (100):
The rule broker component 121 will route a rule query either to the general rules engine or to a sequence of rule authorities until an answer is obtained or no more authorities are available. The rule broker component 121 routes queries based only on the rule name and does not validate the parameter list. The rule authority or authorities are responsible for validating the parameters.

Detailed Description Text (101):
A preferred protocol exists between the rule broker and components querying the rule broker. Any component querying the rule broker must be prepared to handle the case of "no answer" gracefully. For example, a no answer may occur when no such rule is registered or when the component registered to answer the rule cannot answer. Also, the rule broker component 121 must return a specific "no answer" answer to a requester when no answer is available. Further, the rules engine and all rule authorities should dynamically check the parameter list and return the appropriate "no answer" if a discrepancy between the expected and received parameters exist.

Detailed Description Text (103):
A set of complex rules, spanning multiple configuration tables, is used in the AGS implementation when choosing what dispense amounts are displayed on selection buttons to a customer withdrawing cash at a CAT 16. The existing "withdraw cash" application is tightly coupled to the structure of these tables. The acquirer component 114 might register as a rule authority for the "WhatDispenseAmounts?" question. The input parameters for this question include the product type, which specifies the product being withdrawn from, and the currency. The output parameters include the result code and the variable length list of amounts. Some of the session data needed to answer the question, such as card type and level of service, is available from known session components and consequently is not passed as input. The acquirer component 114, in processing the request, may query whatever database contains specific rules for dispense amounts and ask the peripheral device manager component 62 to determine what denominations are available.

Detailed Description Text (105):
As another example, a rule "MaxPICRetries?" to be processed by the rule engine is registered in the rule database. This rule has no input parameters and has output parameters of a result code and MaxPICRetries. As rule data, some interpretable data which indicates that a "business options" table should be searched for the MaxPICRetries value matching the session values of issuer and card type. All of the session data needed to answer the question, such as the issuer and card type, is available from known session components so no specific input parameters are needed. The rule engine searches the specified table for a match on the session issuer and card type and returns the value of MaxPICRetries for that match.

Detailed Description Text (108):
The delivery system 12 is also not limited to any particular integrated development environment (IDE). The IDE, however, should have support for multi-user shared development and should have integration with a configuration management capability. The IDE should also support a tool "plug-in" capability to allow tools to be added which are unique to the delivery system 12. Some examples of these "plug-in" tools include configuration tools to allow for the maintenance of system configuration information

and test tools including host and device emulators. Other tools include software distribution tools to standardize the method by which software upgrades are distributed, system management and logging tools, security protocols, and middle-ware for distributed object support in legacy system interfaces. Further tools include template development tools for both canonical and device specific templates, a rules database editor, services registry maintenance tools, and language man repository editor. The IDE preferably supports all of the selected targeted languages so as to minimize retraining and allows reuse of "plug-in" of tools across development languages. The operating system for the delivery system 12 is preferably Microsoft's Windows NT but may alternatively operate on other operating systems, such as a Macintosh or a UNIX operating system.

Detailed Description Text (109):
A component in the delivery system 12 may comprise any piece of hardware or software that is potentially independently replaceable with the software components being embodied as either executables (.EXEs) or dynamically loaded libraries (.DLLs). Components generally have well-defined interfaces. An application, in contrast, is a set of components that does a specific business function, such as cash withdrawal and may comprise several components. Each application in the delivery system 12 preferably comprises one or more dialog components 83 for handling the user interface, one or more business rule components 121, and one or more transaction executor components 91 for handling the message interface with external service providers 22.

CLAIMS:

11. The system as set forth in claim 1, further comprising a rule broker component for registering a rule authority for each question and for directing requests for an answer to an appropriate rule authority.

12. The system as set forth in claim 11, wherein the mini-app dialog component validates the customer information using the rule broker component.

61. The method as set forth in claim 39, further comprising instantiating a rule broker component, registering rule authorities with the rule broker component, receiving a question at the rule broker, and directing the question to a respective rule authority.

62. The method as set forth in claim 61, further comprising validating customer information with the rule broker component on behalf of the mini-app dialog component.

WEST

☐ Generate Collection Print

L11: Entry 50 of 54 File: USPT Feb 16, 1999

DOCUMENT-IDENTIFIER: US 5873094 A
TITLE: Method and apparatus for automated conformance and enforcement of behavior in application processing systems

Abstract Text (1):
A model information control system ("MICS") is used in conjunction with a user-defined information model and one or more conventional information system program modules or "functions"to execute business applications. Following each execution of an action defined by the information model, an Expected Behavior Control System ("EBCS") determines if the resulting behavior is consistent with expected behavior and conforms the behavior to the expected behavior if necessary. The EBCS may also change the behavior of an application processing system or application program by enforcing a new behavior for the application processing system or application program without modifying them. The MICS includes an event-action-state machine that manipulates the user-defined information model and the functions.

Brief Summary Text (9):
These type of problems normally remain undetected until the application program breaks down or until poor performance is revealed due to unnecessary processing of actions. In a traditional system, the user must trace the processing flow and observe each action within the processing flow to analyze the performance of the system or to identify errors. Thus, it would be advantageous and very useful to have a system which conforms expected behavior or enforces specific behavior of each action of the application processing system and eliminates incorrect or unnecessary processing to optimize system performance.

Brief Summary Text (11):
The present invention overcomes the foregoing and other problems with a method and apparatus for automated conformance and enforcement of the expected behavior in an application processing system. An expected behavior control system ("EBCS") is integrated with an application processing system including at least one application program having actions defined therein. With respect to the EBCS, the terms "action," "rules," "expression" and "function" are synonymous unless otherwise specified. The EBCS includes an expected behavior function for detecting if an executed action by the control program produces with an expected behavior. Additional, the EBCS may conform or enforce selected behavior to modify the process flow of an application processing system method without modifying the application programs. When an expected behavior is achieved, the application processing system executes the next action defined by the application program. If an unexpected behavior is detected, a correction action function executes at least one corrective action in order to conform or enforce the behavior of the system to achieve an expected behavior.

Detailed Description Text (2):
Referring now to FIG. 1, a block diagram is shown of the preferred embodiment of the model information control system (MICS) 10 of the present invention. The MICS 10 includes at least one information model 12 which is created using a text editor and specific semantic rules defined for a target computer system upon which the MICS executes. Several information models are shown in detail below. The target computer system typically comprises a 486-based computer running a UNIX, DOS or OS/2 operating system of the like, although other target systems are useful as well. The MICS also includes a control system engine (CSE) 14 which is preferably a finite state control action machine that cycles through states of operation as will be described in more detail below. The control system engine 14 performs several actions. It activates an object from a set of objects defined in the information model 12. The control system engine 14 also activates an action from a set of actions 16 associated with the object. This is achieved using an associating action flag as will be described.

Detailed Description Text (4):
The information model consists of at least one attribute and one control flag or at least one object and a dictionary of attributes. The object generally will include a set of attributes that constitutes a "subset" of the attributes in the dictionary, a number of control flags, and one or more rules or expressions. The control flags, in conjunction with the expressions, facilitate the CSE's use of the object. Preferably, control flags are divided into four distinct types, namely (1) attribute type flags, (2) object type flags that control storage and retrieval of the object's instant(s), (3) action type flags that facilitate activation and processing of instant(s) through one or more functions, and (4) function control flags used with attributes to provide control of attributes specific to the function, e.g., presentation function control flags that can be passed to a user interface function to control the presentation and/or interactions between the user interface function and the various I/O devices of the system (such as the presentation display and report printer).

Detailed Description Text (6):
The attribute "existence" classes of flags control the rules for creating attributes. For example, ATYPE and "select" attribute constraint expressions specify the conditions on how to create one or more attributes. For example the attribute "09:00 A.M." in the Appointment Calendar Information Model has the control flag "ATYPE=T" which specifies the attribute name is of type time, and the attribute "select" constraint defines how to create another attribute which is explained in detail later.

Detailed Description Text (9):
As noted above, the action type flags in the object identify the action which facilitates activation and processing of instant(s) by the CSE. An action consists of four control flags, namely "action identifier," "instant type," "action class," "instant propagation type," and optionally one or more "function." An action identifier flag is the flag by which the particular action is known external to the system. The instant type flag indicates the type of instant (active or empty). The action class flag identifies which class of actions the particular action belongs. An instant propagation type flag indicates how to propagate the change of instant for a primitive or atomic object of the instant. The functions if defined may be required to be executed to process the instant.

Detailed Description Text (12):
According to a feature of the invention, attributes and objects in the information model use consistent rules or "expressions" that, in conjunction with the various control flags, facilitate the application process flow to execute information model. The attributes of an object contain (i) information on how to assign values to these attributes, (ii) value constraints limiting the kind of values that can be assigned to these attributes, (iii) dependency information specifying how assigning a particular value to one attribute affects the value of another attribute, and (iv) select constraints creating one or more attributes with specific constraints. More specifically, at the attribute level these rules include (i) "initial values" or "default values" expressions that assign values to these attributes, (ii) "values," "condition" or "assignment" expressions limiting the kind of values that can be assigned to these attributes, and (iii) "activation" expressions specifying how to change the value of attributes when instances are activated, (iv) "propagation" expressions specifying how assigning a particular value to one attribute affects the value of another attribute, (v) "select" expressions specifying constraints limiting the kind of instants that can be activated. Similar rules or expressions are implemented at the object level. When used in an object, these expressions modify the behavior of the instant of the object to effect the activation, instantiation and processing of information. For example, a constraint expression at the object level specifies the activation constraint limiting what kind of instants of the object can be activated. Finally, the object also includes index attributes by which instances of an object can be stored and retrieved via the database using the value of one or more attributes of the object.

Detailed Description Text (13):
Thus according to the invention, an object generally includes one or more control flags that facilitate the CSE's use of the object, a subset of attributes selected from a dictionary of attributes, and one or more rules or expressions that modify the behavior of an object's instants to affect the activation, instantiation and processing of information. An object's attributes may also include expressions that control how to assign values to these attributes, that limit the kind of values that can be assigned to these attributes, that specify how to change the value of attributes when instances

are activated, that specify how assigning a particular value to one attribute affects the value of another attribute, and that specify constraints limiting the kind of instants that can be activated.

Detailed Description Text (28):
The first field in the first line includes the object's name (in this case PATIENT CHART). The second field in the line includes the object type flag identifying the type of object (in this case primitive). Following the object type flag, the next field identifies the action type flags that identify the one or more actions that the CSE can select for activation, instantiation and processing of instant(s) of an object. In this example, an object having the DATABASE action class flag merely indicates to the CSE to substitute all actions associated with the DATABASE action class as available actions (thus making it unnecessary to individually list each action of the set).

Detailed Description Text (39):
Prior to the CSE execution, one or more information models are created by a developer using semantic rules for the target system. Each information model has one of three states: idle, active and suspended. When an information model is in its idle state, its "next state" is active. An information model in the active state can be suspended at any time during the processing of the information model. Once suspended, the information model can be returned to its active state (at the same point therein where processing was suspended) or the suspended information model can be placed back into its idle state. Each information model has associated therewith a attribute identifying its state and its next state. Execution of the control system engine begins with State 1. In state 1, the CSE is initialized. This step obtains the necessary resources (e.g., memory, buffer pools, interprocess communication environment, I/O interface environment) from the operating system and then reads the action table definitions and function table definitions and loads them into memory. The CSE also creates structure in the memory to hold the information models, and loads information models into the memory. State 1 also sets up the user environment's presentation interface 26.

Detailed Description Text (45):
In State 6, the routine first determines if there is any function flag associated with the active action. If one or more function flags exist, then for each function flag, the CSE executes the associated function and any process propagation expression associated with the active object over all instants which satisfy activation constraints in the object. The function may be an instantiation function, e.g., a user instantiation function that interacts with the user using the presentation interface and presentation control flags. The CSE will also process any attribute flags by mapping the attribute flags into appropriate functions. For example, the presentation flags are used to create windows and control the interaction with the data in the window. Only data that satisfies attribute constraints is accepted and upon acceptance, the CSE also executes any propagation expression associated with any attribute. If the user decides, however, to access another information model during State 6, the CSE sets the active model's next state equal to 6, suspends the active model and proceeds back to State 2.

Detailed Description Text (50):
As can be seen, the medical system is much more complicated as compared to the simple notepad and calendar applications. Despite the length of the information model itself, however, it should be appreciated that the model is all that is required for purposes of executing the business application. In other words, once the medical system information model is written, it is not required for the user to write source code or other high level code to implement the model. It is further not required that the user maintain source code or the like to implement enhancements or to correct errors. In this regard, all the user need do is to modify the information model (e.g., by adding an attribute, modifying or adding an object, changing an expression in an object, etc.). In operation, the information model is directly processed by the CSE. The MICS may include a preprocessor to process the expressions if desired and the functions can be embedded in the MICS or can be distinct therefrom and thus executed using IPC facilities. The base MICS need not be changed or altered.

Detailed Description Text (82):
Assuming the user enters the key value for PATIENT NAME, the CSE interacts until a valid key value is entered. The CSE also activates instant associated with PATIENT NAME by instantiating PATIENT CHART attributes using the database interface. Assuming activation is successful, the CSE looks for an activation expression associated with PATIENT CHART object. There is none here; if one existed it would have been executed.

Detailed Description Text (133):
Thus a designer desiring to take advantage of the present invention builds models as opposed to applications, thereby allowing end users to create their own applications directly from the business model. As set forth above, a complete medical system is implemented using less than about 700 lines of an information model as compared to thousands upon thousands of lines of complex source code. The main processing engine of the MICS is the CSE machine in which each action creates an event, the event changes a state, and the state triggers an action. A model can be terminated during any state and can be restarted at the same state later. According to the invention, whenever a transaction is terminated, which may occur for example when another model is called for processing, the control engine saves the prior model and the state of the transaction. When the other processing is completed, the control engine reenters the previous transactions at the "action" level (as opposed to the instruction level). The control engine is thus reentrant at the action level.

Detailed Description Text (136):
Referring now to FIG. 3, there is illustrated a preferred embodiment of the present invention. The application processing system 30 consists of an action processor 32 for executing application programs by executing one or more actions required by an application program. The execution of these actions generates a particular behavior. With respect to MICS, the action processor will comprise the control system engine and the application programs will consist of information models. An EBCS 34 interfaces with the action processor 32 and an external input/output 36. In most cases, the EBCS 34 executes after each execution of an application "action" which occurs during active instants (State 5) and process instants (State 6) of an application processing system 32 such as the MICS. However in some cases, execution of the EBCS may occur before execution of an action or expression to check for certain conditions affecting enforcement of a particular process flow. With respect to the ECBS the terms "rules", "expressions" and "actions" are synonymous unless otherwise specified.

Detailed Description Text (137):
The EBCS 34 comprises an expected behavior function (EBF) 38 for conforming or enforcing processing system behavior to expected behavior rules in response to the resulting behavior of an action initiated by the application program and a correction action function 40 for executing one or more corrective actions to conform or enforce the expected behavior if an expected behavior rule is not met. The expected behavior function 38 and correction action function 40 may be incorporated as separate rules or as a single rule without effecting the overall operation of the present invention. The expected behavior function 38 consists of a set of rules for determining whether or not an expected behavior for an executed application program action has occurred. This rule may either conform the behavior of the system to an expected behavior or enforce the processing flow of the actions in a desired manner. If the conditions set by the rules within the EBF 38 are not met, a correction ID number is returned which the correction action function 40 uses to enable an associated response for executing actions to achieve the expected behavior for the application processing system.

Detailed Description Text (138):
The expected behavior rules of the EBF verify specific behavior for actions initiated by the application programs and implement corrective actions for conforming or enforcing the expected behavior of the system when necessary. The expected behavior rules are divided into two groups, missing action rules and invalid action rules. The missing action rules search for specific behavior which should result in response to a particular action. The invalid action rules detect and correct behavior occurring which should not be occurring. A sample of these rules for conforming system behavior are as follows:

Detailed Description Text (139):
Missing action rules

Detailed Description Text (142):
Invalid action rules

Detailed Description Text (145):
The above described rules are not meant to be limiting, and any number of rules for conforming and correcting the expected behavior of the system may be used.

Detailed Description Text (148):
Case 1: Assume a user has selected the ADD action of the BANK ACCOUNT object. In execution of State 6 by the CSE, the database operation fails because ACT ID index key

attribute value is indexed but the value assignment for ACT ID never occurs because the field is protected from data entry by the FPROT attribute flag and there is no propagation expression or "action" to assign to the attribute value. Upon execution of the EBCS, conformance with the EBCS missing action rule 1 is not achieved, and the correction action function executes "correct specification to assign index key attribute value." Here EBCS correction actions of the correction action function can automatically remove the FPROT attribute flag from ACT ID attribute definition of BANK ACCOUNT to correct the problem without having the user modify the information model, or EBCS correction actions can interact with the user through the input/output interface by displaying the attribute definition and asking the user to change the flag or add a propagation expression to correct the problem to enforce the expected behavior.

Detailed Description Text (149):
Case 2: In the TRANSACTION object the JOURNAL flag instructs the CSE to activate the JOURNAL object of TRANSACTION after a database operation. Assume that the user has selected the ADD action of TRANSACTION. If the AMOUNT attribute value is not entered by the user, then the CSE fails to create the JOURNAL database record because upon execution of the JOURNAL object, no attribute value is changed because the propagation expression only changes attribute values if AMOUNT is not zero. EBCS conformance missing action rule number 2 is not achieved, and the correction action function executes "correct specification to create cross reference." The correction action can enforce the behavior by automatically adding the MUST FILL flag to the attribute definition AMOUNT of TRANSACTION or EBCS correction actions can interact with the user using input/output interface to ask the user to change the MUST FILL flag or modify the propagation expression of the JOURNAL object to change at least one attribute value.

Detailed Description Text (150):
Case 3: Assume that the user has selected the ADD action and the user enters a CODE value equal to ICR. The REFERENCE value then changes to 999999. This causes the AMOUNT value to become negative, which in turn causes the REFERENCE value to change to 999999. This creates a loop condition and conformance with the EBCS invalid rule 2 fails, and the EBCS executes "terminate loop condition". The correction action function can conform the behavior by automatically terminating the loop condition or asking the user to do so via the input/output interface.

Detailed Description Text (153):
Presently when an application requests use of a database record for updating purposes, the application initiates a READ UPDATE action that reads the entry within the database and locks the record into an UPDATE mode. During the READ UPDATE action, no other user may access the database record. This is due to the fact that the READ UPDATE action initiated by the application program generates a processing flow that only enables one user to access the desired database. A new type of processing flow may be enforced by the EBCS by using a set of enforcement rules similar to the conformance rules discussed previously. The enforcement rules are different from the conformance rules in that they may be executed both before and after the execution of an action. This is due to the fact that the expected behavior for the action may be different depending upon the existing conditions.

Detailed Description Text (154):
Assume the following rules are also included within the EBCS:

Detailed Description Text (157):
Thus when a READ UPDATE action is executed by the application processing system the EBCS rule 1 causes the record to be read without update. This enables more than one user on the network or client/server computing system to simultaneously update the same record without locking the record during an update session. Upon completion of each user's modifications to a record, each user's updates are returned to the database in accordance with rule 2. Rule number 1 is a pre-enforcement rule executed before execution of an action to direct the processing flow in a desired direction without altering the application programs. In this case, the rule initiates only a READ action. Rule number 2 is a post-enforcement rule for enforcing a particular behavior after execution of an action. In this case, the rule reads the database record for UPDATE READ and modifies the record with the changes made by the user.

Detailed Description Text (158):
Rules such as this enable multiple users to update a database record as long as different attributes of the database record are being updated. In a normal processing system, whatever changes are last made to the attribute values of a record are the changes that are finally used within the database record. Should a conflict occur

between multiple users trying to update the same attribute during READ UPDATE actions, additional rules may be included in the EBCS to enable resolution of the conflict. A rule may be included in the EBCS enforcing that whichever user has entered the largest number of attribute value changes within a specific group gets priority for all attribute value changes for that group. Thus, in the case where a first user updates one attribute in the record of a first patient and five attributes within the record of a second patient, and a second user only updates ten attributes within a record of the second patient, the corrective actions force the changes made by the first user to control and be entered into the first patient's record, and the attributes entered by the second user to control and be entered into the second patient's record.

Detailed Description Text (159):
It should be noted from the foregoing discussion that none of the application programs are being altered in any way. All that is altered is the manner in which the process flow of the actions is carried out to enable new system behaviors to be achieved without reprogramming the application programs. It should also be noted that the particular rules discussed for enforcing process flow are merely illustrative and that any number of rules may be used to achieve any desired behavior.

Detailed Description Paragraph Table (2):
______________________________________ SYSTEM CONTROL TABLES
______________________________________ #ACTION ADD,EMPTY INSTANT,DATABASE,UPDATE INSTANT,USR INSTANTIATIONUPDATE,ACTIVE INSTANT,DATABASE,UPDATE INSTANT,USR INSTANTIATION DELETE,ACTIVE INSTANT,DATABASE,DELETE INSTANT, USR.sub.-- INSTANTIATION REPORT,ACTIVE INSTANT,REPORT,0,0, CREATE.sub.-- REPORT CREATE ATOMIC DATA,ACTIVE INSTANT, ANALYSIS,UPDATE INSTANT,CREATE.sub.-- ATOMIC.sub.-- DATA GRAPH,ACTIVE INSTANT,ANALYSIS,0,0,GRAPH LEARN ASSOCIATION,ACTIVE INSTANT,ANALYSIS,0,0, LEARN.sub.-- ATOMIC.sub.-- DATA USE ASSOCIATION,ACTIVE INSTANT, ANALYSIS,0,0,USE.sub.-- ATOMIC.sub.-- DATA #MODEL APPOINTMENT CALENDAR,PRIMITIVE,UPDATE,LEARN Description You can use Calendar program to keep track of your appointments SELECT, INSTANT (NEXT INSTANT) [DATE] += 1; (PREV INSTANT) [DATE] -= 1. DATE,TYPE=D,UNIQUE DATE, TYPE=D,PTYPE=D,TEMP 09:00 A.M.,TYPE=C,LEN=66,ATYPE=t Description Type any information you want to include for this time slot. SELECT,ATTRIBUTE (ATTRIBUTE < 5.30) ATTRIBUTE += 60. NOTE,TYPE=C,LEN=132 Description Type up to two lines of text as notes. TODO,TYPE=C,LEN=132 Description Type up to two lines of text for things to do. #END MODEL NOTEPAD,PRIMITIVE, UPDATE, LEARN DATE,TYPE=D,GROUP=100,SETDATE,FPROT,UNIQUE TIME,TYPE=T,GROUP=100,SETTIME,FPROT REFERENCE,TYPE=A,LEN=20,GROUP=100 TEXT,TYPE=C,LEN=512,GROUP=101,PHIDE #END MODEL MEDICAL #OBJECT PATIENT CHART,PRIMITIVE,DATABASE ACNT NO,TYPE=1,LEN=8,GROUP=100,FPROT DEFAULT VALUE [ACNT NO] = SEQNUMBER. VISIT FIRST,TYPE=d,GROUP=100,FPROT,SETDATE LAST,TYPE=D,GROUP=100,FPROT,SETDATE PATIENT NAME,TYPE=A,LEN=30,GROUP=102,MFILL ADDRESS,TYPE=A,LEN=30,GROUP=103,MFILL CITY,TYPE=A,LEN=15,GROUP=104,MFILL STATE,TYPE=A,LEN=2,GROUP=104,MFILL ZIP CODE,TYPE=L,PTYPE=Z,LEN=5,GROUP=104,MFILL BIRTHDATE,TYPE=D,GROUP=105,MFILL SEX,TYPE=A,LEN=1,GROUP=105,MFILL Constraint,values M - MALE, F - FEMALE. MARITAL STATUS,TYPE=A,LEN=1,GROUP=105,MFILL Constraint,values S - SINGLE, M - MARRIED, W - WIDOWED, D - DIVORCED. PHONE NO(HOME),TYPE=S,PTYPE=P,LEN=10,GROUP=106,RJUST (OFFICE),TYPE=S,PTYPE=P,LEN=10,GROUP=106,RJUST SOCIAL SEC.#,TYPE=L,PTYPE=S,LEN=9,GROUP=107 DRIVER LIC.#,TYPE=C,LEN=15,GROUP=107 CREDIT CARD#,TYPE=C,PTYPE=N,LEN=16,GROUP=108 TYPE,TYPE=A,LEN=10,GROUP=108 Full time student,P - Part time student. OCCUPATION,TYPE=A,LEN=20,GROUP=109 NAME,TYPE=A,LEN=30,GROUP=111 RELATIONSHIP,TYPE=A,LEN=1,GROUP=112 Constraint,values S - self,s - spouse,c - child,o - other. PHONE NO(HOME),GROUP=112,DATA=1,RJUST (OFFICE),GROUP=112,DATA=1,RJUST OCCUPATION,GROUP=113,DATA=1 EMPLOYER'S NAME,GROUP=113,DATA=1 RESP. PARTY'S NAME,TYPE=A,LEN=30,GROUP=115 RELATIONSHIP,GROUP=115,DATA=1 BIRTHDATE,GROUP=116,DATA=1 SEX,GROUP=116,DATA=1 SOCIAL SEC.#,GROUP=116,DATA=1 DRIVER LIC.#,GROUP=117,DATA=1 CREDIT CARD#,GROUP=117,DATA=1 TYPE,GROUP=117,DATA=1 PHONE NO(HOME),GROUP=118,DATA=2,RJUST (OFFICE),GROUP=118,DATA=2,RJUST ADDRESS,GROUP=119,DATA=1 CITY,GROUP=120,DATA=1 STATE,GROUP=120,DATA=1 ZIP CODE,GROUP=120,DATA=1 OCCUPATION,GROUP=121,DATA=2 EMPLOYER'S NAME,GROUP=121,DATA=2 INSURED?,TYPE=A,LEN=1,GROUP=122 Constraint,values Y - YES, N - NO. CARRIER CODE,GROUP=122 PLAN NAME,TYPE=A,LEN=20,GROUP=123 INS ID,TYPE=C,LEN=12,GROUP=123 GROUP ID,TYPE=C,LEN=12,GROUP=123 OTHER INSURANCE?,TYPE=A,LEN=1,GROUP=125 Constraint,values Y - YES, N - NO. CARRIER CODE,DATA=1,GROUP=125 PLAN NAME,GROUP=126,DATA=1 INS ID,GROUP=126,DATA=1 GROUP ID,GROUP=126,DATA=1 INSURED'S NAME,TYPE=A,LEN=30,GROUP=127 RELATIONSHIP,TYPE=A,LEN=1,GROUP=127,DATA=2 BIRTHDATE,TYPE=D,GROUP=128,DATA=2 SEX,GROUP=128,DATA=2 EMPLOYER'S NAME,GROUP=128,DATA=3 ACTIVE STMNT,TYPE=I,LEN=2,GROUP=130 Default value [ACTIVE STMNT] =1. BALANCE,TYPE=R,LEN=10,DEC=2,GROUP=130,FPROT OB DUE,TYPE=D,GROUP=130

RECALL,TYPE=D,GROUP=130 CHARGES,TYPE=R,LEN=10,DEC=2,TEMP,FHIDE POST AMOUNTS,TYPE=R,LEN=10,DEC=2,TEMP,FHIDE CREDITS,TYPE=R,LEN=10,DEC=2,TEMP,FHIDE PATIENT PORTION,TYPE=R,LEN=10,DEC=2,GROUP=131 INSURANCE COVERAGE(%),TYPE=R,LEN=10,DEC=2,GROUP=131 Propagation [PATIENT PORTION] = (100 - [INSURANCE COVERAGE(%)]) *.01 * ([BALANCE] -[DEDUCTIBLE]); [PATIENT PORTION] = [PATIENT PORTION] + [DEDUCTIBLE]. DEDUCTIBLE,TYPE=R,LEN=10,DEC=2,GROUP=131 BILLING DATE,TYPE=D,GROUP=132 INS BILLING DATE,TYPE=D,GROUP=132 PATIENT PAYMENTS,TYPE=R,LEN=10,DEC=2,GROUP=133 INSURANCE PAYMENTS,TYPE=R,LEN=10,DEC=2,GROUP=133 INSURANCE TYPE,TYPE=A,LEN=1,GROUP=134 Constraint,values E - MEDICARE,D - MEDICAID,G - GROUP HEALTH PLAN. AUTHORIZATION NO,TYPE=C,LEN=12,GROUP=134 MEDICAID RESUB CODE,TYPE=C,LEN=12,FHIDE ORIGINAL REF NO,TYPE=C,LEN=16,FHIDE DATE OF SYMPTOM (CURRENT),TYPE=D,FHIDE (PRIOR),TYPE=D,FHIDE REFERRING PHYSICIAN,TYPE=A,LEN=20,FHIDE PHYSICIAN ID,TYPE=C,LEN=10,FHIDE FACILITY,TYPE=A,LEN=20,FHIDE HOSPITAL DATE (FROM),TYPE=D,FHIDE (TO),TYPE=D,FHIDE UNBL TO WORK (FROM),TYPE=D,GROUP=140,FHIDE (TO),DATA=1,FHIDE OUTSIDE LAB?,TYPE=A,LEN=1,FHIDE Constraint,values Y - YES, N - NO. LAB AMNT,TYPE=R,LEN=8,DEC=2,FHIDE NICOE NAME,TYPE=A,LEN=25,GROUP=137 DESCRIPTION ENTER THE NAME OF PERSON TO BE NOTIFIED IN CASE OF EMERGENCY. NICOE PHONE NO,GROUP=137 CONDITION RELATED TO JOB?,TYPE=A,LEN=1,FHIDE Constraint,values Y - YES, N - NO. AUTO ACCIDENT? ,TYPE=A,LEN=1,FHIDE Constraint,values Y - YES, N - NO. OTHER ACCIDENT?,TYPE=A,LEN=1,FHIDE Constraint,values Y - YES, N - NO. #OBJECT BILLING DATA,PRIMITIVE,JOURNAL.sub.-- RCD,HIDE ACNT NO,DPROT STMNT,DPROT TRANS DATE,TYPE=D Constraint,condition ([TRANS DATE] LE DATE). Description Enter date when service started for this procedure. TO DATE,TYPE=D Constraint,condition ([TO DATE] LE DATE); ([TO DATE] GE [TRANS DATE]). Description Enter date when service ended for this procedure in case of surgery or pregnancy to and from dates may be different. PS,TYPE=A,LEN=1. Constraint,values IH - in patient hospital,OH - out patient hospital, O- office. TS,TYPE=N,LEN=1 Constraint,values 1 - medical care,2 - surgery,3 - consultation,4 -xray,5 laboratory, 8 -assistant surgery,9 - other medical service, 0 - blood or packed red shell. CODE,TYPE=A,LEN=4,UNIQUE Description Enter four byte alias procedure code or payment/adjust.codes: payments - pca(cash), pck(check), pin(insurance) ,pcl(collection) :payment error adjust. - cpe(credit postingerror) or dpe(debit): payment - dop(refund), drc(returncheck), drc(return check fee): dlf(legal/court), dfc(financecharge): charges adjustment - cad(credit) or dda(debit) orccd(discount). Propagation [AMOUNT] = [AMNT]; ([UNIT] EQUAL TO 0,) [UNIT] = 1. DIAG,TYPE=A,LEN=6 Description Enter valid diagnosis code from icd 9 code book. UNIT,TYPE=I,LEN=2 AMOUNT,TYPE=R,LEN=8,DEC=2 DESCRIPTION,TEMP AMOUNTS,FHIDE,TEMP OLD AMOUNT,TYPE=R,LEN=8,DEC=2,FHIDE,TEMP POST AMNT,TYPE=R,LEN=8,DEC=2,FHIDE,TEMP #OBJECT BILLING DATABASE,VIEW of BILLING DATA, DATABASE Initial values SETVALUE([STMNT],[ACTIVE STMNT]). activation propagation [OLD AMOUNT] = [AMOUNT] * [UNIT] * [CDFLG]. Propagation [POST AMNT] = [AMOUNT] * [UNIT] * [CDFLG] - [OLDAMOUNT]; [POST AMOUNTS] = [POST AMOUNTS] + [POST AMNT]; (([CTYPE] EQUAL TO 9) AND ([POST AMNT] > 0)) [PATIENT PAYMENTS] = [PATIENT PAYMENTS] + [POST AMNT]; (([CTYPE] EQUAL TO 8) AND ([POST AMNT] > 0)) [INSURANCE PAYMENTS] = [INSURANCE PAYMENTS]+ [POST AMNT]; [LAST] = DATE; ([STMNT] > [ACTIVE STMNT]) [ACTIVE STMNT] = [STMNT]. link propagation ([POST AMOUNTS] != 0) [BALANCE] = [BALANCE] + [POSTAMOUNTS]. DOCNAME,FHIDE,HEADER,DPROT LINE1,FHIDE,HEADER,DPROT LINE2,FHIDE,HEADER,DPROT LINE3,FHIDE,HEADER,DPROT LINE4,FHIDE,HEADER,DPROT ACNT STRING,PHIDE,COL=50,GROUP=98,FPROT,DPROT PATIENT NAME,GROUP=100,FPROT,PHIDE,COL=5,DPROT ACNT NO,GROUP=100,FPROT,COL=55,DPROT ADDRESS,GROUP=101,FPROT,PHIDE,COL=5,DPROT STMNT,GROUP=101,FPROT,COL=57,DPROT CITY,GROUP=102,FPROT,PHIDE,COL=5,DPROT STATE,GROUP=102,FPROT,PHIDE,FCAT,DPROT ZIP CODE,GROUP=102,FPROT,PHIDE,FCAT,DPROT

Detailed Description Paragraph Table (6):

#MODEL PAYMENT MANAGER #OBJECT BANK ACCOUNT,ADD,UPDATE,REPORT ACT ID,UNIQUE,FPROT NAME ACCOUNT NO BALANCE #OBJECT TRANSACTION,ADD,UPDATE,REPORT,JOURNAL propagation ([ACT ID] NEQ "") [PAYEE] = [NAME] . CODE propagation ([CODE] IS "ICR") [REFRENCE] = 999999 . DATE PAYEE AMOUNT propagation ([AMOUNT<0) [REFERENCE] = 999999 . REFERENCE propagation ([REFERENCE] IS "999999") [AMOUNT] -=[AMOUNT] . ACT ID propagation [PAYEE] = [NAME] . STATUS # TRANSACTION JOURNAL,JOURNAL OF TRANSACTION,HIDE ([AMOUNT] is not "0") [CTYPE] = [CODE] . CTYPE PAYEE DATE AMOUNT ACT ID #BALANCE CHECKBOOK, VIEW OF TRANSACTION, BROWSE, REPORT select,ALL ([STATUS] is "O") . activation ([LAST BALANCE] IS ZERO) [LAST BALANCE] = [BALANCE] ; [LAST BALANCE] -= [AMOUNT] . propagation [BALANCE] -= [AMOUNT] . CODE,ENTRY=15 PAYEE DATE AMOUNT LAST BALANCE,ENTRY=1 #INDEX TRANSACTION,ACT ID

Other Reference Publication (2):

H. Wang, et al., "A Framework for Handling Errors during the Execution of Trigger Rules for an Active Object-Oriented DBMS," Proc. Fourth Int'l. Workshop on Research Issues in

Data Engineering, pp. 132-136, Feb. 1994.

CLAIMS:

1. A method for automatically detecting and correcting a behavior of an information model, the information model defining a process flow for a software application and being executable in a process flow control engine to perform actions having expected behaviors, comprising the steps of:

monitoring a behavior associated with execution of a given action in an information model;

comparing the monitored behavior associated with execution of the given action to an expected behavior rule defining an expected behavior associated with the action in the information model to determine if the monitored behavior reflects the expected behavior of the information model; and

executing a corrective action when the monitored behavior does not follow the expected behavior rule to correct the monitored behavior associated with the given action of the information model to reflect the expected behavior of the information model.

2. The method of claim 1, wherein the expected behavior rule defines a behavior that should result from execution of the given action of the information model.

3. The method of claim 1, wherein the expected behavior rule defines a behavior that should not result from execution of the given action of the information model.

4. A method for automatically detecting and correcting a behavior of an information model, the information model defining a process flow for a software application and being executable in a process flow control engine to perform actions having expected behaviors, comprising the steps of:

monitoring a behavior associated with execution of a set of one or more actions of the information model as the information model is executed by the process flow control engine;

comparing the monitored behavior associated with execution of the set of one or more actions of the information model to an expected behavior rule defining an expected behavior of the information model to detect differences between the monitored behavior of the information model and the expected behavior of the information model; and

executing a corrective action within the information model to correct the monitored behavior to reflect the expected behavior when the monitored behavior of the information model does not follow the expected behavior rule.

5. An expected behavior control system for detecting and correcting a behavior of an information model, the information model defining a process flow for a software application and being executable in a process flow control engine to perform actions having expected behaviors, comprising:

an expected behavior function operative as the information model is executed by the process flow control engine for detecting if an action executed by the information model provides an expected behavior in accordance with at least one expected behavior rule defining the expected behavior resulting from the executed action of the information model; and

a corrective action function operative as the information model is executed by the process flow control engine for executing at least one corrective action on the information model to correct behavior of the information model in response to detection of unexpected behavior of the information model by the expected behavior function.

6. The system of claim 5 wherein the expected behavior function includes a plurality of rules defining expected behaviors associated with actions executed by the information model.

7. The system of claim 6 wherein the corrective action function includes at least one corrective action corresponding to one of the plurality of rules defining an expected behavior.

8. The system of claim 6 wherein the plurality of rules are divided into two groups, comprising:

missing action rules for defining invalid behaviors associated with the execution of an action; and

invalid action rules for defining invalid behaviors associated with the execution of an action.

9. The system of claim 5 wherein the expected behavior function includes at least one rule for enforcing a particular process flow in the information model.

10. An application processing control system for executing at least one software application, comprising:

an information model executable in a process flow control engine defining a single process flow for at least one software application using at least one action; and

an expected behavior control system, responsive to execution of the information model, comprising:

a database defining expected behavior rules associated with the execution of the at least one action;

an expected behavior function operative as the information model is executed by the process flow control engine for detecting if an executed action within the information model provides an expected behavior in accordance with at least one expected behavior rule defining the behavior associated with the executed action of the information model; and

a corrective action function operative as the information model is executed by the process flow control engine for executing at least one corrective action with the information model to correct behavior in response to detection of unexpected behavior by the expected behavior function.

11. The system of claim 10 wherein the expected behavior control system further comprises:

an expected behavior function operative as the information model is executed by the process flow control engine for detecting if an action executed by the information model provides an expected result in accordance with at least one expected behavior rule defining the behavior resulting from an executed action; and

a corrective action function operative as the information model is executed by the process flow control engine for executing at least one corrective action with the information model to correct behavior in response to detection of unexpected behavior by the expected behavior function.

12. The system of claim 11 wherein the expected behavior function includes a plurality of rules defining expected behaviors associated with actions executed by the information model.

13. The system of claim 12 wherein the corrective action function includes at least one corrective action corresponding to one of the plurality of rules defining an expected behavior.

14. The system of claim 12 wherein the plurality of rules are divided into two groups, comprising:

missing action rules for defining invalid behaviors associated with the execution of an action; and

invalid action rules for defining invalid behaviors associated with the execution of an action.

15. The system of claim 11 wherein the expected behavior function includes at least one rule for enforcing a particular process flow in the information model.

18. The system of claim 16 wherein the expected behavior control system further

comprises:

an expected behavior function operative as the information model is executed by the process flow control engine for detecting if an action executed by the information model provides an expected behavior in accordance with at least one expected behavior rule defining the behavior resulting from an executed action; and

a corrective action function operative as the information model is executed by the process flow control engine for executing at least one corrective action in the information model to correct behavior of the information model in response to detection of unexpected behavior by the expected behavior function.

19. The system of claim 18 wherein the expected behavior function includes at least one rule for enforcing a particular process flow in the information model.